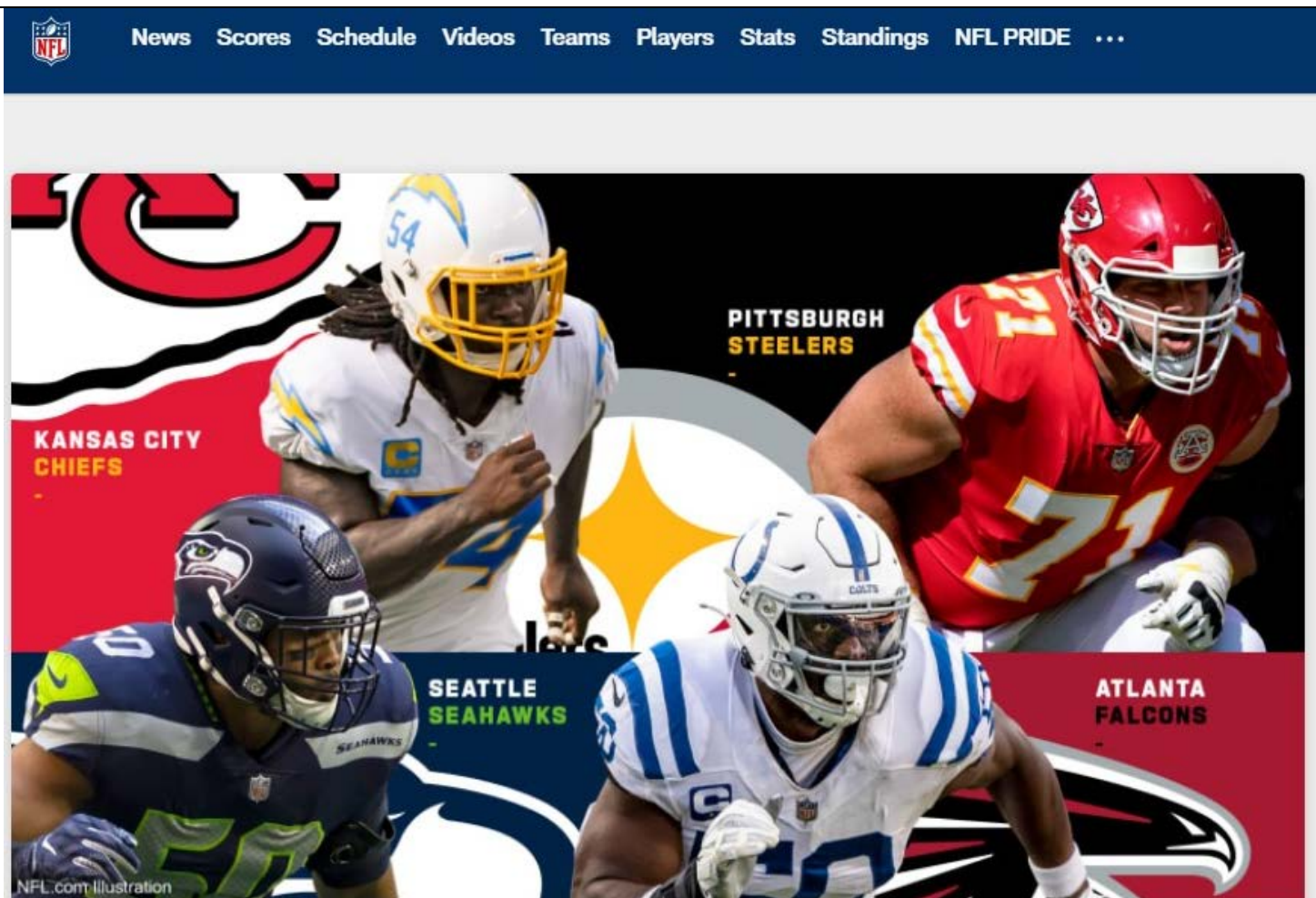
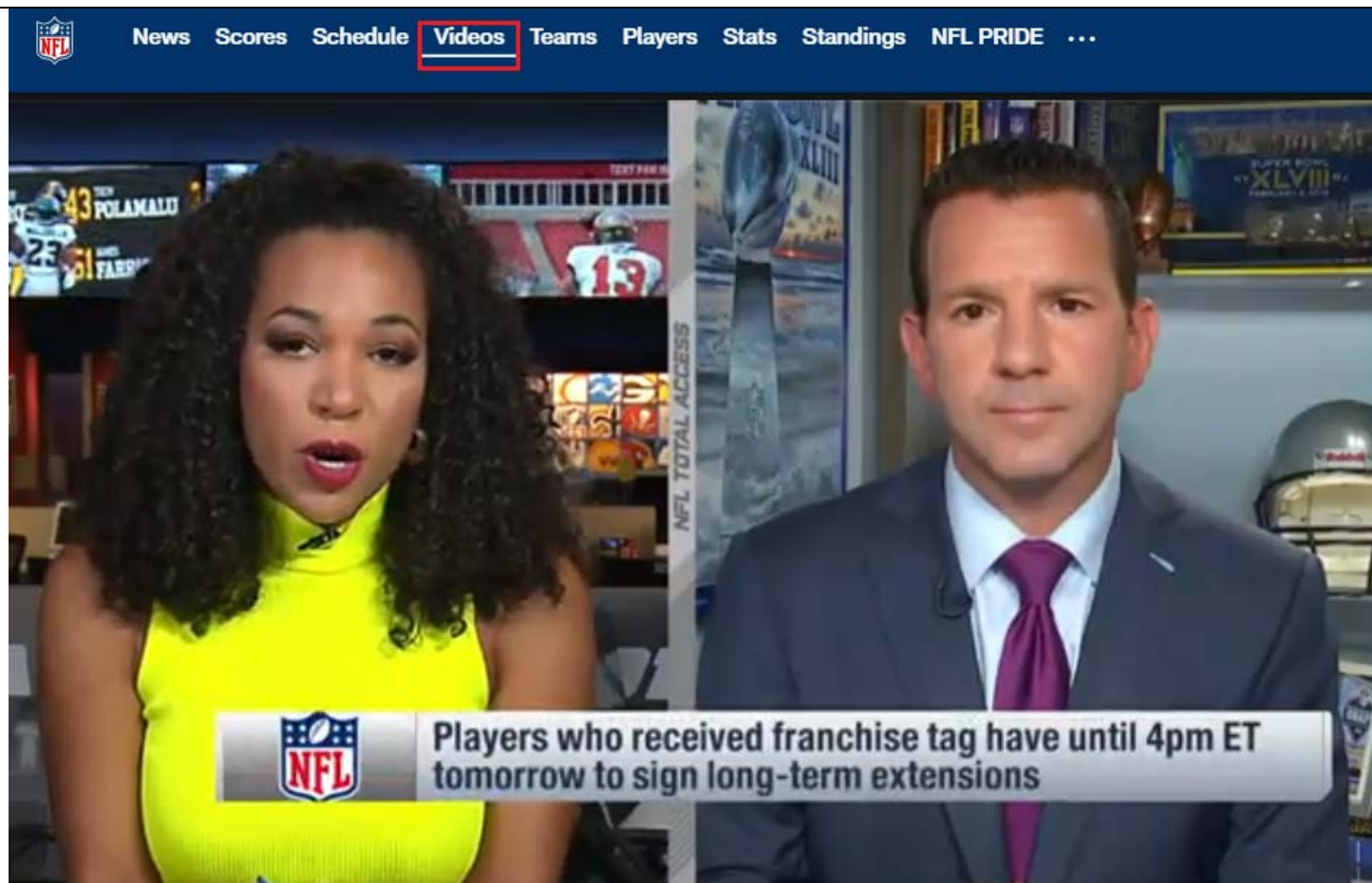


EXHIBIT D

US8195828B2	NFL.com
<p>1. A method for discontinuous transmission, in sections, of encoded video data in a network of distributed appliances, the method comprising the following steps:</p>	<p>NFL.com support HLS streaming protocol (“the Standard”), including for delivery of contents to its customers/viewers, including in its internal testing and usage.</p> <p>On information and belief, Defendant performs all steps of claim 1 or, alternatively, to the extent a viewer performs any step, Defendant conditions the viewer’s use of the Defendant’s accused instrumentalities using the Standard on the performance of that step as disclosed herein. For example, on information and belief, a viewer cannot use the accused instrumentality utilizing the Standard as described in this claim chart without performance of the steps recited in this claim. By providing the accused instrumentality utilizing the Standard as disclosed herein, Defendant also controls the manner and/or timing of the functionality described in this claim chart. In other words, for a viewer to utilize the functionality described in this claim chart, the steps of this claim must be performed in the manner described herein. Without performance of the steps as described herein, the Defendant’s functionality will not be available to viewers.</p> <p>As shown below, a video content from NFL.com is streamed and the data traffic is captured showing the media format (HLS), the m3u8 file, (e.g., the Media playlist file comprising links to content chunks in .ts format used by HLS to contain information about the media playing), and the encryption scheme used by the streamed video. In addition, the HLS stream provided through NFL.com provides trick mode operation (such as 10 second reverse and forward trick modes) to the streamed video.</p> <p>The Standard practices a method for discontinuous transmission (e.g., discontinues transmission of video data during a trick mode such as fast-forward, fast-rewind, etc. in which an I-frame version is transmitted), in sections (e.g., media segments), of encoded video data (e.g., Codec encoded representations of video stream) in a network of distributed appliances (e.g., network of HLS distribution servers and client devices).</p>



<https://www.nfl.com/>



<https://www.nfl.com/videos/>

Shown below is the URL of .m3u8 master file sent by the NFL.com server, which identifies the usage of HLS based streaming by NFL.com servers. The .m3u8 master file refers to all the variants of the video encoded for various bandwidths and resolutions. The URL of .m3u8 master file is: <https://dcs-vod.apis.anvato.net/vod/p/session/master.m3u8?i=i177471499-n91f2d446-9e01-448b-a085->

4ea04cb8c83d&anvtrid=w5741a7f3ae79f3c1fe4d8caa729b689, which redirects to <https://au5e6f8g7ryz8sqdsuw3dcdapdf.ctl.nfl.com> for content streaming.

The screenshot displays the Fiddler tool interface. The top toolbar includes buttons for 'Get Started', 'Statistics', 'Inspectors', 'AutoResponder', 'Composer', 'Fiddler Orchestra Beta', and 'FiddlerScript'. Below the toolbar, a tabbed interface shows 'Headers', 'TextView', 'SyntaxView', 'WebForms', 'HexView', 'Auth', 'Cookies', 'Raw', 'JSON', and 'XML'. The 'Headers' tab is active, showing the 'Request Headers' section. The request is a GET to [/vod/p/session/master.m3u8?i=177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741a7f3ae79f3c1fe4d8caa729b689](https://vod/p/session/master.m3u8?i=177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741a7f3ae79f3c1fe4d8caa729b689) with HTTP/1.1. The 'Client' section lists headers: Accept: */*, Accept-Encoding: gzip, deflate, br, Accept-Language: en-US,en;q=0.9, and User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36. The 'Miscellaneous' section shows the Referer: https://www.nfl.com/. The 'Security' section shows the Origin: https://www.nfl.com, sec-ch-ua: "Not;A Brand";v="99", "Google Chrome";v="91", "Chromium";v="91", and sec-ch-ua-mobile: ?0. Below the request headers, the 'Response Headers' section shows HTTP/1.1 200 OK. The 'Cache' section lists Cache-Control: max-age=0, no-cache, no-store, Date: Thu, 15 Jul 2021 04:17:55 GMT, and Vary: Accept-Encoding. The 'Entity' section lists Content-Length: 4400 and Content-Type: application/x-mpegURL.

Request Headers [Raw] [Header Definitions]

GET /vod/p/session/master.m3u8?i=177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741a7f3ae79f3c1fe4d8caa729b689 HTTP/1.1

Client

Accept: */*
 Accept-Encoding: gzip, deflate, br
 Accept-Language: en-US,en;q=0.9
 User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36

Miscellaneous

Referer: https://www.nfl.com/

Security

Origin: https://www.nfl.com
 sec-ch-ua: "Not;A Brand";v="99", "Google Chrome";v="91", "Chromium";v="91"
 sec-ch-ua-mobile: ?0

Response Headers [Raw] [Header Definitions]

HTTP/1.1 200 OK

Cache

Cache-Control: max-age=0, no-cache, no-store
 Date: Thu, 15 Jul 2021 04:17:55 GMT
 Vary: Accept-Encoding

Entity

Content-Length: 4400
 Content-Type: application/x-mpegURL

Packet Captured by Fiddler tool

Headers	TextView	SyntaxView	WebForms	HexView	Auth	Cookies	Raw	JSON	XML			
Request Headers [Raw] [Header Definitions] GET /vod/p/ <u>master.m3u8?encp=FZUTWdlRPOCbFQ29evfVww.pY6I5k6P5FQzr0xl60F1EbuNs6_kbFsVBMnGcSSWP79J-OebtWgVr9CA3J65wrsaxUs1pYEWY8i6SYDRscxPm</u>												
Client Accept: */* Accept-Encoding: gzip, deflate, br Accept-Language: en-US,en;q=0.9 User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36												
Miscellaneous Referer: <u>https://www.nfl.com/</u>												
Security Origin: https://www.nfl.com sec-ch-ua: "Not;A Brand";v="99", "Google Chrome";v="91", "Chromium";v="91" sec-ch-ua-mobile: ?0												
Transformer	Headers	TextView	SyntaxView	ImageView	HexView	WebView	Auth	Caching	Cookies	Raw	JSON	XML
<pre>{ "session_id": "1177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d", "master_m3u8": "<u>https://dcs-vod.apis.anvato.net/vod/vp/session/master.m3u8?i=1177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741a7f3ae79f3c1fe4d8caa729b689</u>", "interstitials": { "cues": [], "content": { "output_duration": 69.036 }, "breaks": [], "info": {} } }</pre>												

Packet Captured by Fiddler tool

#	Result	Protocol	Host	URL	Content-Type
135	200	HTTP	Tunnel to	apv-static.minute.ly:443	
136	304	HTTPS	access-prod.apis.a...	/anvacks/GVvEgwyJeWe...	
137	206	HTTPS	apv-static.minute.ly	/videos/v-7174b7e0-b125...	video/mp4
138	200	HTTP	Tunnel to	tkx.apis.anvato.net:443	
139	200	HTTP	Tunnel to	apv-static.minute.ly:443	
140	200	HTTPS	tkx.apis.anvato.net	/rest/v2/server_time?anv...	application/json
141	200	HTTPS	tkx.apis.anvato.net	/rest/v2/mcp/video/94445...	application/x-javasc.
142	200	HTTP	Tunnel to	dcs-vod.apis.anvato.net...	
143	206	HTTPS	apv-static.minute.ly	/videos/v-7174b7e0-b125...	video/mp4
144	200	HTTPS	dcs-vod.apis.anvat...	/vod/p/master.m3u8?enc...	application/json
145	206	HTTPS	apv-static.minute.ly	/videos/v-7174b7e0-b125...	video/mp4
146	304	HTTPS	www.nfl.com	/compiledassets/assets/fo...	
147	200	HTTP	Tunnel to	p.nfltags.com:443	
148	206	HTTPS	apv-static.minute.ly	/videos/v-7174b7e0-b125...	video/mp4
149	206	HTTPS	p.nfltags.com	/nfl/fonts/icons/NFLIcons...	application/font-woff
150	200	HTTP	Tunnel to	content-autofill.googleapi...	
151	206	HTTPS	p.nfltags.com	/nfl/fonts/icons/NFLIcons...	application/font-woff
152	200	HTTPS	content-autofill.goo...	/v1/pages/CHRDaHvbw...	text/plain
153	200	HTTP	Tunnel to	dcs-vod.apis.anvato.net...	
154	200	HTTPS	dcs-vod.apis.anvat...	/vod/p/session/master.m3...	application/x-mpeg.
155	200	HTTP	Tunnel to	player-health.apis.anvato...	
156	200	HTTP	Tunnel to	player-health.apis.anvato...	
157	200	HTTP	Tunnel to	dpm.demdex.net:443	
158	200	HTTP	Tunnel to	nfl.hb.omtrdc.net:443	
159	200	HTTPS	dcs-vod.apis.anvat...	/vod/p/355400/prog.m3u...	application/x-mpeg..
160	200	HTTP	Tunnel to	au5e6f8g7yz8sqdswu3dc...	

Log

Filters

Timeline

Get Started

Statistics

Inspectors

AutoResponder

Composer

Fiddler Orchestra Beta

FiddlerScript

Headers

TextView

SyntaxView

WebForms

HexView

Auth

Cookies

Raw

JSON

XML

Request Headers

GET /vod/p/session/master.m3u8?i=177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741a7f3ae79f3c1fe4d8caa729b689 HTTP/1.1

Client

Accept: */*

Accept-Encoding: gzip, deflate, br

Accept-Language: en-US,en;q=0.9

User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36

Miscellaneous

Referer: https://www.nfl.com/

Security

Origin: https://www.nfl.com

sec-ch-ua: "Not;A Brand";v="99", "Google Chrome";v="91", "Chromium";v="91"

sec-ch-ua-mobile: ?0

Transformer

Headers

TextView

SyntaxView

ImageView

HexView

WebView

Auth

Caching

Cookies

Raw

JSON

XML

Response Headers

HTTP/1.1 200 OK

Cache

Cache-Control: max-age=0, no-cache, no-store

Date: Thu, 15 Jul 2021 04:17:55 GMT

Vary: Accept-Encoding

Entity

Content-Length: 4400

Content-Type: application/x-mpegURL

Packet Captured by Fiddler tool

No.	Time	Method	Host	Path	Content-Type	Status
143	206	HTTPS	apv-static.minute.ly	/videos/v-7174b7e0-b125...	video/mp4	
144	200	HTTPS	dcs-vod.apis.anvat...	/vod/p/master.m3u8?enc...	application	
145	206	HTTPS	apv-static.minute.ly	/videos/v-7174b7e0-b125...	video/mp4	
146	304	HTTPS	www.nfl.com	/compiledassets/assets/fo...		
147	200	HTTP	Tunnel to	p.nfltags.com:443		
148	206	HTTPS	apv-static.minute.ly	/videos/v-7174b7e0-b125...	video/mp4	
149	206	HTTPS	p.nfltags.com	/nfl/fonts/icons/NFLIcons...	application	
150	200	HTTP	Tunnel to	content-autofill.googleapi...		
151	206	HTTPS	p.nfltags.com	/nfl/fonts/icons/NFLIcons...	application	
152	200	HTTPS	content-autofill.goo...	/v1/pages/ChrDah3vbw...	text/plain	
153	200	HTTP	Tunnel to	dcs-vod.apis.anvato.net:...		
154	200	HTTPS	dcs-vod.apis.anvat...	/vod/p/session/master.m3...	application	
155	200	HTTP	Tunnel to	player-health.apis.anvato...		
156	200	HTTP	Tunnel to	player-health.apis.anvato...		
157	200	HTTP	Tunnel to	dpm.demdex.net:443		
158	200	HTTP	Tunnel to	nfl.hb.omtrdc.net:443		
159	200	HTTPS	dcs-vod.apis.anvat...	/vod/p/355400/prog.m3u...	application	
160	200	HTTP	Tunnel to	au5e6f9g7ryz8sqdsuw3dc...		
161	200	HTTP	Tunnel to	ad16a8c8b860d62e9d55b...		
162	200	HTTPS	player-health.apis....	/pixel.png?player_type=...	image/png	
163	200	HTTPS	player-health.apis....	/pixel.png?player_type=...	image/png	
164	200	HTTPS	dpm.demdex.net	/dd?id_tbd=json&id_ver=2...	application	
165	200	HTTP	Tunnel to	gpizacfw3gtfw4vmh5nc4u...		
166	200	HTTPS	nfl.hb.omtrdc.net	/settings/f75c3025512d2c...	application	
167	200	HTTP	Tunnel to	r7c2k6b3.ssl.hwcdn.net:443		
168	200	HTTP	Tunnel to	gpizacfw3gtfw4vmh5nc4u...		
169	200	HTTP	Tunnel to	gpizacfw3gtfw4vmh5nc4u...		
170	200	HTTP	Tunnel to	gpizacfw3gtfw4vmh5nc4u...		
171	200	HTTP	Tunnel to	gpizacfw3gtfw4vmh5nc4u...		
172	200	HTTP	Tunnel to	gpizacfw3gtfw4vmh5nc4u...		
173	200	HTTPS	ad16a8c8b860d62e...	/0/wsg	application	
174	200	HTTP	Tunnel to	storage.googleapis.com:443		

Request Headers

GET /vod/p/session/master.m3u8?i=177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741a73ae79f3c1fe4d8caa729b689 HTTP/1.1

Client

Accept: */*
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36

Miscellaneous

Referer: https://www.nfl.com/

Security

Origin: https://www.nfl.com
sec-ch-ua: "Not A Brand";v="99", "Google Chrome";v="91", "Chromium";v="91"
sec-ch-ua-mobile: ?0

Transformer Headers TextView SyntaxView ImageView HexView WebView Auth Caching Cookies Raw JSON XML

#EXTM3U
#EXT-X-INDEPENDENT-SEGMENTS
#EXT-X-STREAM-INF: BANDWIDTH=355400, FRAME-RATE=30, RESOLUTION=416x234, AVERAGE-BANDWIDTH=308000, CODECS="mp4a.40.2, avc1.42
https://dcs-vod.apis.anvato.net/vod/p/355400/prog.m3u8?i=177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741
#EXT-X-STREAM-INF: BANDWIDTH=502400, FRAME-RATE=30, RESOLUTION=416x234, AVERAGE-BANDWIDTH=448000, CODECS="mp4a.40.2, avc1.42
https://dcs-vod.apis.anvato.net/vod/p/502400/prog.m3u8?i=177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741
#EXT-X-STREAM-INF: BANDWIDTH=691400, FRAME-RATE=30, RESOLUTION=640x360, AVERAGE-BANDWIDTH=628000, CODECS="mp4a.40.2, avc1.42
https://dcs-vod.apis.anvato.net/vod/p/691400/prog.m3u8?i=177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741
#EXT-X-STREAM-INF: BANDWIDTH=1006400, FRAME-RATE=30, RESOLUTION=640x360, AVERAGE-BANDWIDTH=928000, CODECS="mp4a.40.2, avc1.4
https://dcs-vod.apis.anvato.net/vod/p/1006400/prog.m3u8?i=177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741
#EXT-X-STREAM-INF: BANDWIDTH=1426400, FRAME-RATE=30, RESOLUTION=768x432, AVERAGE-BANDWIDTH=1328000, CODECS="mp4a.40.2, avc1.
https://dcs-vod.apis.anvato.net/vod/p/1426400/prog.m3u8?i=177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741
#EXT-X-STREAM-INF: BANDWIDTH=2266400, FRAME-RATE=30, RESOLUTION=768x432, AVERAGE-BANDWIDTH=2128000, CODECS="mp4a.40.2, avc1.
https://dcs-vod.apis.anvato.net/vod/p/2266400/prog.m3u8?i=177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741
#EXT-X-STREAM-INF: BANDWIDTH=3526400, FRAME-RATE=30, RESOLUTION=1280x720, AVERAGE-BANDWIDTH=3328000, CODECS="mp4a.40.2, avc1
https://dcs-vod.apis.anvato.net/vod/p/3526400/prog.m3u8?i=177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741
#EXT-X-STREAM-INF: BANDWIDTH=5416400, FRAME-RATE=30, RESOLUTION=1280x720, AVERAGE-BANDWIDTH=5128000, CODECS="mp4a.40.2, avc1
https://dcs-vod.apis.anvato.net/vod/p/5416400/prog.m3u8?i=177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741
#EXT-X-STREAM-INF: BANDWIDTH=8566400, FRAME-RATE=30, RESOLUTION=1920x1080, AVERAGE-BANDWIDTH=8128000, CODECS="mp4a.40.2, avc
https://dcs-vod.apis.anvato.net/vod/p/8566400/prog.m3u8?i=177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741
#EXT-X-T-FRAME-STREAM-INF: BANDWIDTH=308000, RESOLUTION=416x234, CODECS="avc1.42000d, mp4a.40.2", URI="https://dcs-vod.apis
#EXT-X-I-FRAME-STREAM-INF: BANDWIDTH=448000, RESOLUTION=416x234, CODECS="avc1.42000d, mp4a.40.2", URI="https://dcs-vod.apis
#EXT-X-I-FRAME-STREAM-INF: BANDWIDTH=628000, RESOLUTION=640x360, CODECS="avc1.42001e, mp4a.40.2", URI="https://dcs-vod.apis
#EXT-X-I-FRAME-STREAM-INF: BANDWIDTH=928000, RESOLUTION=640x360, CODECS="avc1.4d001e, mp4a.40.2", URI="https://dcs-vod.apis
#EXT-X-I-FRAME-STREAM-INF: BANDWIDTH=1328000, RESOLUTION=768x432, CODECS="avc1.4d001e, mp4a.40.2", URI="https://dcs-vod.apis
#EXT-X-I-FRAME-STREAM-INF: BANDWIDTH=2128000, RESOLUTION=768x432, CODECS="avc1.4d001e, mp4a.40.2", URI="https://dcs-vod.apis

Packet Captured by Fiddler tool

143	206	HTTPS	apv-static.minute.ly	/videos/v-7174b7e0-b125...	video/mp4
144	200	HTTPS	dcs-vod.apis.avnat...	/vod/p/master.m3u8?enc...	applicator
145	206	HTTPS	apv-static.minute.ly	/videos/v-7174b7e0-b125...	video/mp4
146	304	HTTPS	www.nfl.com	/compiledassets/assets/fo...	
147	200	HTTP	Tunnel to	p.nftags.com:443	
148	206	HTTPS	apv-static.minute.ly	/videos/v-7174b7e0-b125...	video/mp4
149	206	HTTPS	p.nftags.com	/nfl/fonts/icons/NFLIcons...	applicator
150	200	HTTP	Tunnel to	content-autofill.googleapp...	
151	206	HTTPS	p.nftags.com	/nfl/fonts/icons/NFLIcons...	applicator
152	200	HTTPS	content-autofill.goo...	/v1/pages/ChrDAJHvBwV...	text/plain
153	200	HTTP	Tunnel to	dcs-vod.apis.avnato.net:...	
154	200	HTTPS	dcs-vod.apis.avnat...	/vod/p/session/master.m3...	applicator
155	200	HTTP	Tunnel to	player-health.apis.avnato...	
156	200	HTTP	Tunnel to	player-health.apis.avnato...	
157	200	HTTP	Tunnel to	dpm.demdex.net:443	
158	200	HTTP	Tunnel to	nfl.hb.omtrdc.net:443	
159	200	HTTPS	dcs-vod.apis.avnat...	/vod/p/355400/prog.m3u...	applicator
160	200	HTTP	Tunnel to	au56f8g7ryz8sqdsuw3dc...	
161	200	HTTP	Tunnel to	ad16a8cb8b60d62e9d55b...	
162	200	HTTPS	player-health.apis...	/pixel.png?player_type=...	image/png
163	200	HTTPS	player-health.apis...	/pixel.png?player_type=...	image/png
164	200	HTTPS	dpm.demdex.net	/d/d7_rbd=json&id_ver=2...	applicator
165	200	HTTP	Tunnel to	gizpacfw3gtfw4vmh5nc4u...	
166	200	HTTPS	nfl.hb.omtrdc.net	/settings/f75c3025512d2c...	applicator
167	200	HTTP	Tunnel to	r7c2k6b3.ssl.hwcdn.net:443	
168	200	HTTP	Tunnel to	gizpacfw3gtfw4vmh5nc4u...	
169	200	HTTP	Tunnel to	gizpacfw3gtfw4vmh5nc4u...	
170	200	HTTP	Tunnel to	gizpacfw3gtfw4vmh5nc4u...	
171	200	HTTP	Tunnel to	gizpacfw3gtfw4vmh5nc4u...	
172	200	HTTP	Tunnel to	gizpacfw3gtfw4vmh5nc4u...	
173	200	HTTPS	ad16a8cb8b60d62e...	/0/wsg	applicator
174	200	HTTP	Tunnel to	storage.googleapis.com:443	

```
#Request Headers
GET /vod/p/355400/prog_m3u8?i=177471499n91f2d446-9e01-448b-a085-4ea04cb8c83d&nvrid=w5741a73ae79f3c1fe4d8caa729b689 HTTP/1.1

Client
Accept: */*
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36

Miscellaneous
Referer: https://www.nfl.com/

Security
Origin: https://www.nfl.com
sec-ch-ua: "Not A Brand";v="99", "Google Chrome";v="91", "Chromium";v="91"
sec-ch-ua-mobile: ?0
```

Transformer	Headers	TextView	SyntaxView	ImageView	HexView	WebView	Auth	Caching	Cookies	Raw	JSON	XML
	#EXTM3U #EXT-X-VERSION:3 #EXT-X-TARGETDURATION:10 #EXT-X-PLAYLIST-TYPE:VOD #ANAVTO-CDN-PROVIDER: level3 #ANAVTO-SEGMENT-INFO: type=master #EXTINF:9.009, https://aus6ef8q7rvz8sqdsuw3dcdapdf.ct1.nfl.com/league/5684/21/07/14/944452/EBF32C3E8185DAC767B7A44F894861986A95AE75856 #ANAVTO-SEGMENT-INFO: type=master #EXTINF:9.009, https://aus6ef8q7rvz8sqdsuw3dcdapdf.ct1.nfl.com/league/5684/21/07/14/944452/EBF32C3E8185DAC767B7A44F894861986A95AE75856 #ANAVTO-SEGMENT-INFO: type=master #EXTINF:9.009, https://aus6ef8q7rvz8sqdsuw3dcdapdf.ct1.nfl.com/league/5684/21/07/14/944452/EBF32C3E8185DAC767B7A44F894861986A95AE75856 #ANAVTO-SEGMENT-INFO: type=master #EXTINF:9.009, https://aus6ef8q7rvz8sqdsuw3dcdapdf.ct1.nfl.com/league/5684/21/07/14/944452/EBF32C3E8185DAC767B7A44F894861986A95AE75856 #ANAVTO-SEGMENT-INFO: type=master #EXTINF:9.009, https://aus6ef8q7rvz8sqdsuw3dcdapdf.ct1.nfl.com/league/5684/21/07/14/944452/EBF32C3E8185DAC767B7A44F894861986A95AE75856 #ANAVTO-SEGMENT-INFO: type=master #EXTINF:9.009, https://aus6ef8q7rvz8sqdsuw3dcdapdf.ct1.nfl.com/league/5684/21/07/14/944452/EBF32C3E8185DAC767B7A44F894861986A95AE75856 #ANAVTO-SEGMENT-INFO: type=master #EXTINF:9.009,											

Packet Captured by Fiddler tool

The NFL.com server sends references of all the variants of the stream as follows.

```
#EXTM3U
#EXT-X-INDEPENDENT-SEGMENTS
#EXT-X-STREAM-INF: BANDWIDTH=355400, FRAME-RATE=30, RESOLUTION=416x234, AVERAGE-BANDWIDTH=308000, CODECS="mp4a.40.2, avc1.42e00d"
https://dcs-vod.apis.anvato.net/vod/p/355400/prog.m3u8?i=1177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741a7f3
#EXT-X-STREAM-INF: BANDWIDTH=502400, FRAME-RATE=30, RESOLUTION=416x234, AVERAGE-BANDWIDTH=448000, CODECS="mp4a.40.2, avc1.42e00d"
https://dcs-vod.apis.anvato.net/vod/p/502400/prog.m3u8?i=1177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741a7f3
#EXT-X-STREAM-INF: BANDWIDTH=691400, FRAME-RATE=30, RESOLUTION=640x360, AVERAGE-BANDWIDTH=628000, CODECS="mp4a.40.2, avc1.42e01e"
https://dcs-vod.apis.anvato.net/vod/p/691400/prog.m3u8?i=1177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741a7f3
#EXT-X-STREAM-INF: BANDWIDTH=1006400, FRAME-RATE=30, RESOLUTION=640x360, AVERAGE-BANDWIDTH=928000, CODECS="mp4a.40.2, avc1.4d401"
https://dcs-vod.apis.anvato.net/vod/p/1006400/prog.m3u8?i=1177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741a7f3
#EXT-X-STREAM-INF: BANDWIDTH=1426400, FRAME-RATE=30, RESOLUTION=768x432, AVERAGE-BANDWIDTH=1328000, CODECS="mp4a.40.2, avc1.4d401"
https://dcs-vod.apis.anvato.net/vod/p/1426400/prog.m3u8?i=1177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741a7f3
#EXT-X-STREAM-INF: BANDWIDTH=2266400, FRAME-RATE=30, RESOLUTION=768x432, AVERAGE-BANDWIDTH=2128000, CODECS="mp4a.40.2, avc1.4d401"
https://dcs-vod.apis.anvato.net/vod/p/2266400/prog.m3u8?i=1177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741a7f3
#EXT-X-STREAM-INF: BANDWIDTH=3526400, FRAME-RATE=30, RESOLUTION=1280x720, AVERAGE-BANDWIDTH=3328000, CODECS="mp4a.40.2, avc1.4d401"
https://dcs-vod.apis.anvato.net/vod/p/3526400/prog.m3u8?i=1177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741a7f3
#EXT-X-STREAM-INF: BANDWIDTH=5416400, FRAME-RATE=30, RESOLUTION=1280x720, AVERAGE-BANDWIDTH=5128000, CODECS="mp4a.40.2, avc1.4d401"
https://dcs-vod.apis.anvato.net/vod/p/5416400/prog.m3u8?i=1177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741a7f3
#EXT-X-STREAM-INF: BANDWIDTH=8566400, FRAME-RATE=30, RESOLUTION=1920x1080, AVERAGE-BANDWIDTH=8128000, CODECS="mp4a.40.2, avc1.4d401"
https://dcs-vod.apis.anvato.net/vod/p/8566400/prog.m3u8?i=1177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741a7f3
#EXT-X-I-FRAME-STREAM-INF: BANDWIDTH=308000, RESOLUTION=416x234, CODECS="avc1.42000d, mp4a.40.2", URI="https://dcs-vod.apis.anvato.net/vod/p/308000/prog.m3u8?i=1177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741a7f3"
#EXT-X-I-FRAME-STREAM-INF: BANDWIDTH=448000, RESOLUTION=416x234, CODECS="avc1.42000d, mp4a.40.2", URI="https://dcs-vod.apis.anvato.net/vod/p/448000/prog.m3u8?i=1177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741a7f3"
#EXT-X-T-FRAME-STREAM-INF: BANDWIDTH=628000, RESOLUTION=640x360, CODECS="avc1.42001e, mp4a.40.2", URI="https://dcs-vod.apis.anvato.net/vod/p/628000/prog.m3u8?i=1177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741a7f3"
```

Packet Captured by Fiddler tool

The NFL.com server streams video using AVC1 codec (e.g., video encoding) as shown below.

```
X-Anvato-Node: 177471593,177471499
Vary: Accept-Encoding
Via: 1.1 google
Alt-Svc: clear
Content-Length: 4400
```

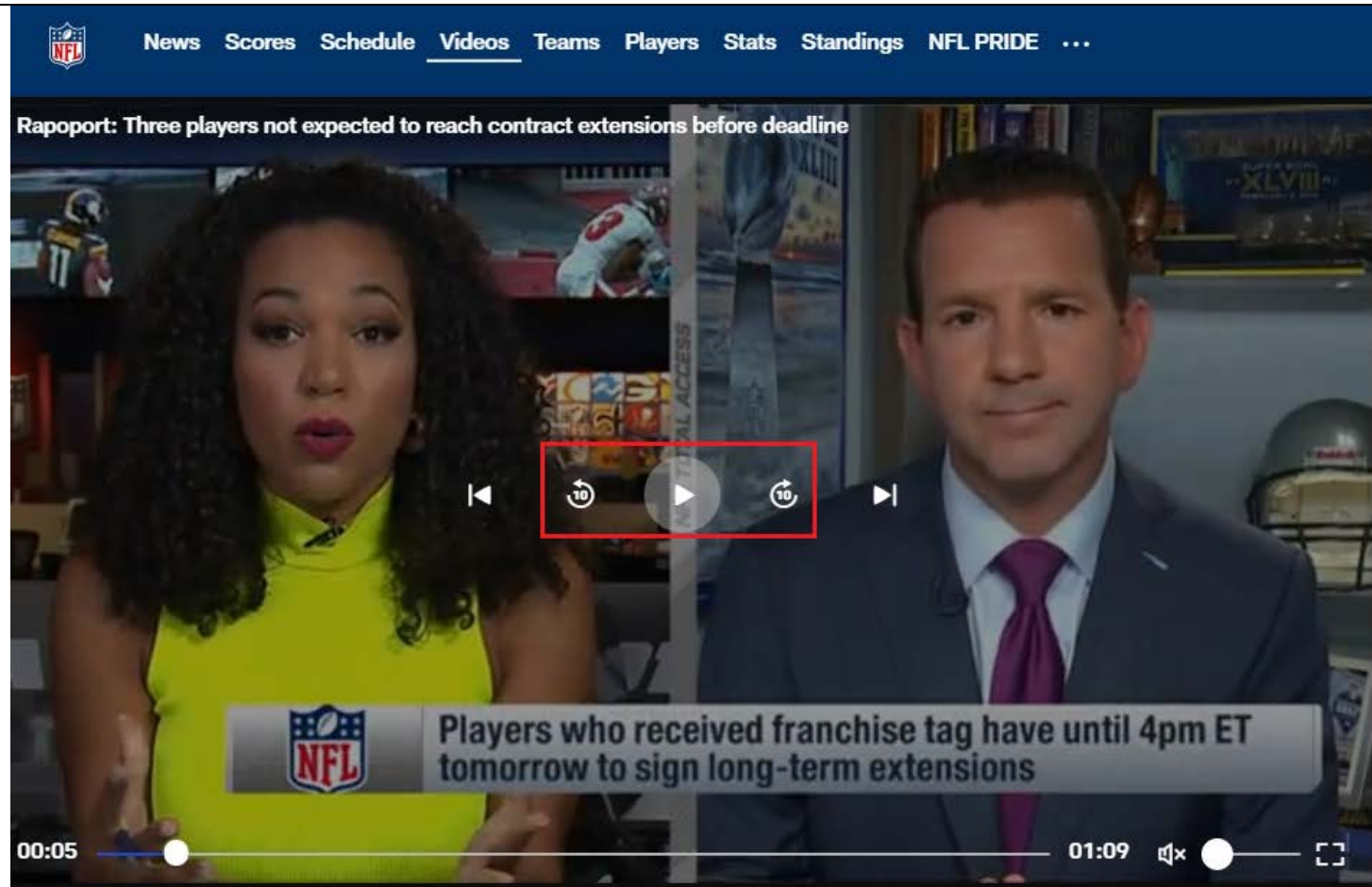
Video encoding

```
#EXTM3U
#EXT-X-INDEPENDENT-SEGMENTS
#EXT-X-STREAM-INF: BANDWIDTH=355400, FRAME-RATE=30, RESOLUTION=416x234, AVERAGE-BANDWIDTH=308000, CODECS="mp4a.40.2, avc1.42e00d"
https://dcs-vod.apis.anvato.net/vod/p/355400/prog.m3u8?i=1177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741a7f3ae79f3c1fe4d
#EXT-X-STREAM-INF: BANDWIDTH=502400, FRAME-RATE=30, RESOLUTION=416x234, AVERAGE-BANDWIDTH=448000, CODECS="mp4a.40.2, avc1.42e00d"
https://dcs-vod.apis.anvato.net/vod/p/502400/prog.m3u8?i=1177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741a7f3ae79f3c1fe4d
#EXT-X-STREAM-INF: BANDWIDTH=691400, FRAME-RATE=30, RESOLUTION=640x360, AVERAGE-BANDWIDTH=628000, CODECS="mp4a.40.2, avc1.42e01e"
https://dcs-vod.apis.anvato.net/vod/p/691400/prog.m3u8?i=1177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741a7f3ae79f3c1fe4d
#EXT-X-STREAM-INF: BANDWIDTH=1006400, FRAME-RATE=30, RESOLUTION=640x360, AVERAGE-BANDWIDTH=928000, CODECS="mp4a.40.2, avc1.4d401e"
https://dcs-vod.apis.anvato.net/vod/p/1006400/prog.m3u8?i=1177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741a7f3ae79f3c1fe4d
#EXT-X-STREAM-INF: BANDWIDTH=1426400, FRAME-RATE=30, RESOLUTION=768x432, AVERAGE-BANDWIDTH=1328000, CODECS="mp4a.40.2, avc1.4d401e"
https://dcs-vod.apis.anvato.net/vod/p/1426400/prog.m3u8?i=1177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741a7f3ae79f3c1fe4d
#EXT-X-STREAM-INF: BANDWIDTH=2266400, FRAME-RATE=30, RESOLUTION=768x432, AVERAGE-BANDWIDTH=2128000, CODECS="mp4a.40.2, avc1.4d401e"
https://dcs-vod.apis.anvato.net/vod/p/2266400/prog.m3u8?i=1177471499-n91f2d446-9e01-448b-a085-4ea04cb8c83d&anvtrid=w5741a7f3ae79f3c1fe4d
#EXT-X-STREAM-INF: BANDWIDTH=3526400, FRAME-RATE=30, RESOLUTION=1280x720, AVERAGE-BANDWIDTH=3328000, CODECS="mp4a.40.2, avc1.4d401f"
```

Packet Captured by Fiddler tool

The NFL.com server sends references (URLs) of all the chunks of the video to be streamed.

The screenshot displays the Fiddler tool interface. The left pane shows a list of HTTP requests. The 159th request is selected, showing details in the right pane. The 'Request Headers' section is expanded, showing various headers. The 'Client' section shows the user agent. The 'Miscellaneous' section shows the referer and security headers. The 'Security' section shows the origin and security headers. A red box highlights the 'Referer' header: 'https://www.nfl.com/'. Another red box highlights the 'sec-ch-ua' header: 'NotA Brand';v=99', 'Google Chrome';v=91', 'Chromium';v=91'. A third red box highlights the 'sec-ch-ua-mobile' header: '0'.



<https://www.nfl.com/videos/>

HTTP Live Streaming

Overview Examples FairPlay Streaming

HTTP Live Streaming

Send live and on-demand audio and video to iPhone, iPad, Mac, Apple Watch, Apple TV, and PC with HTTP Live Streaming (HLS) technology from Apple. Using the same protocol that powers the web, HLS lets you deploy content using ordinary web servers and content delivery networks. HLS is designed for reliability and dynamically adapts to network conditions by optimizing playback for the available speed of wired and wireless connections.

<https://developer.apple.com/streaming/>

HTTP Streaming Architecture

HTTP Live Streaming allows you to send live or prerecorded audio and video, with support for encryption and authentication, from an ordinary web server to any device running iOS 3.0 or later (including iPad and Apple TV), or any computer with Safari 4.0 or later installed.

Overview

Conceptually, HTTP Live Streaming consists of three parts: the server component, the distribution component, and the client software.

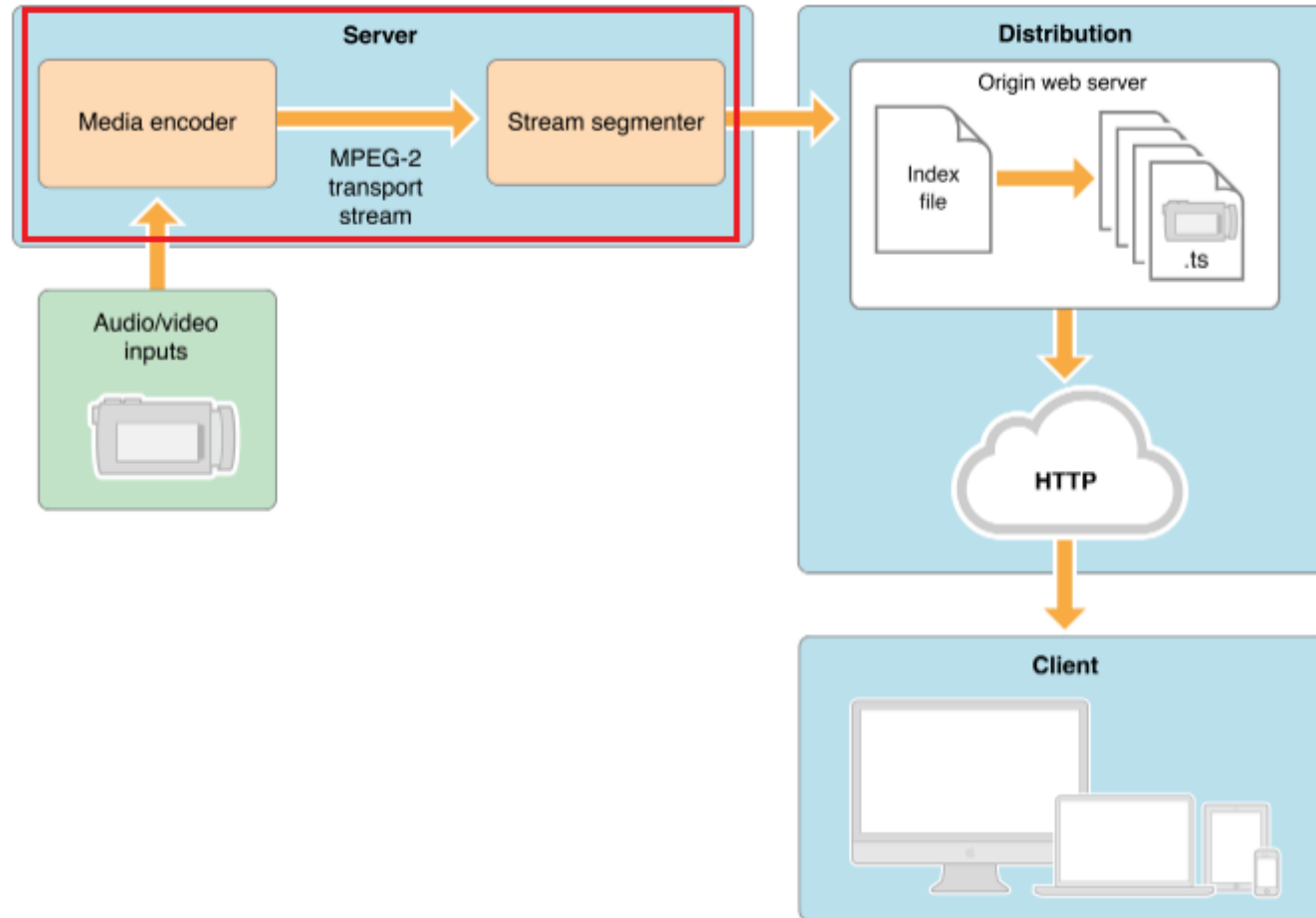
The **server component** is responsible for taking input streams of media and encoding them digitally, encapsulating them in a format suitable for delivery, and preparing the encapsulated media for distribution.

The **distribution component** consists of standard web servers. They are responsible for accepting client requests and delivering prepared media and associated resources to the client. For large-scale distribution, edge networks or other content delivery networks can also be used.

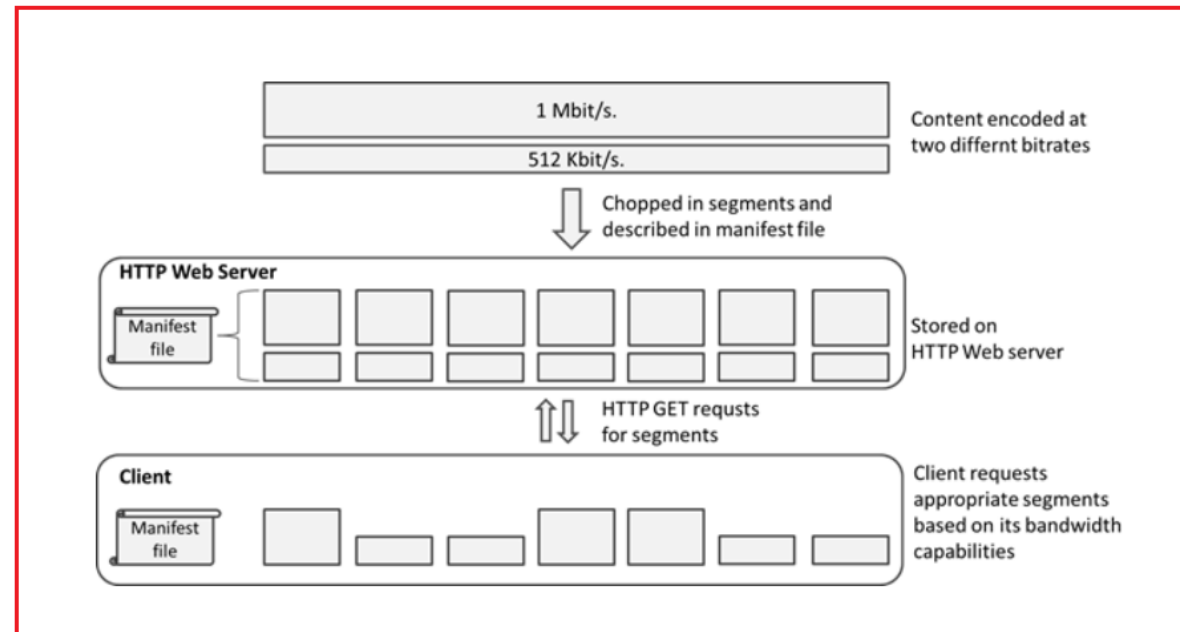
The **client software** is responsible for determining the appropriate media to request, downloading those resources, and then reassembling them so that the media can be presented to the user in a continuous stream. Client software is included on iOS 3.0 and later and computers with Safari 4.0 or later installed.

In a typical configuration, a hardware encoder takes audio-video input, encodes it as H.264 video and AAC audio, and outputs it in an MPEG-2 Transport Stream, which is then broken into a series of short media files by a software stream segmenter. These files are placed on a web server. The segmenter also creates and maintains an index file containing a list of the media files. The URL of the index file is published on the web server. Client software reads the index, then requests the listed media files in order and displays them without any pauses or gaps between segments.

https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/HTTPStreamingArchitecture/HTTPStreamingArchitecture.html#/apple_ref/doc/uid/TP40008332-CH101-SW4



https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/HTTPStreamingArchitecture/HTTPStreamingArchitecture.html#//apple_ref/doc/uid/TP40008332-CH101-SW4



<http://blog.jpauli.tech/live-video-streaming-stack-from-home/>

6.3.3. Playing the Media Playlist File

The client SHALL choose which Media Segment to play first from the Media Playlist when playback starts. If the EXT-X-ENDLIST tag is not present and the client intends to play the media normally, the client SHOULD NOT choose a segment that starts less than three target durations from the end of the Playlist file. Doing so can trigger playback stalls.

Normal playback can be achieved by playing the Media Segments in the order that they appear in the Playlist. The client MAY present the available media in any way it wishes, including normal playback, random access, and trick modes.

The encoding parameters for samples in a Media Segment and across multiple Media Segments in a Media Playlist SHOULD remain consistent. However, clients SHOULD deal with encoding changes as they are encountered, for example, by scaling video content to accommodate a resolution change. If the Variant Stream includes a RESOLUTION attribute, clients SHOULD display all video within a rectangle with the same proportions as that resolution.

<https://tools.ietf.org/html/rfc8216>

Fast Forward and Reverse Playback

HLS supports fast forward and reverse playback through the use of an I-frame playlist. The I-frame playlist points to a byte range within already existing media segments. Fast forward and reverse playback do not need special media segments.

For in-depth information on I-frame playlists, watch [WWDC 2012: Effective HLS](#).

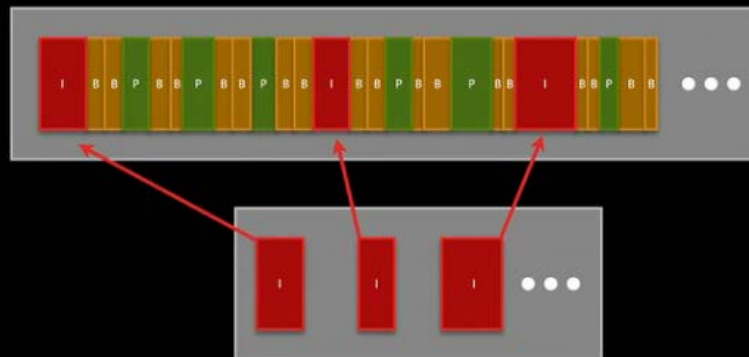
<https://developer.apple.com/library/archive/referencelibrary/GettingStarted/AboutHTTPLiveStreaming/about/about.html>

Fast Forward and Reverse Playback

- Uses I-frame playlists
- Essential for AirPlay and AppleTV
- The higher the I-frame frequency, the smoother the playback

I-frame Playlists

Can reuse existing media files



▶ 24:03 / 41:03

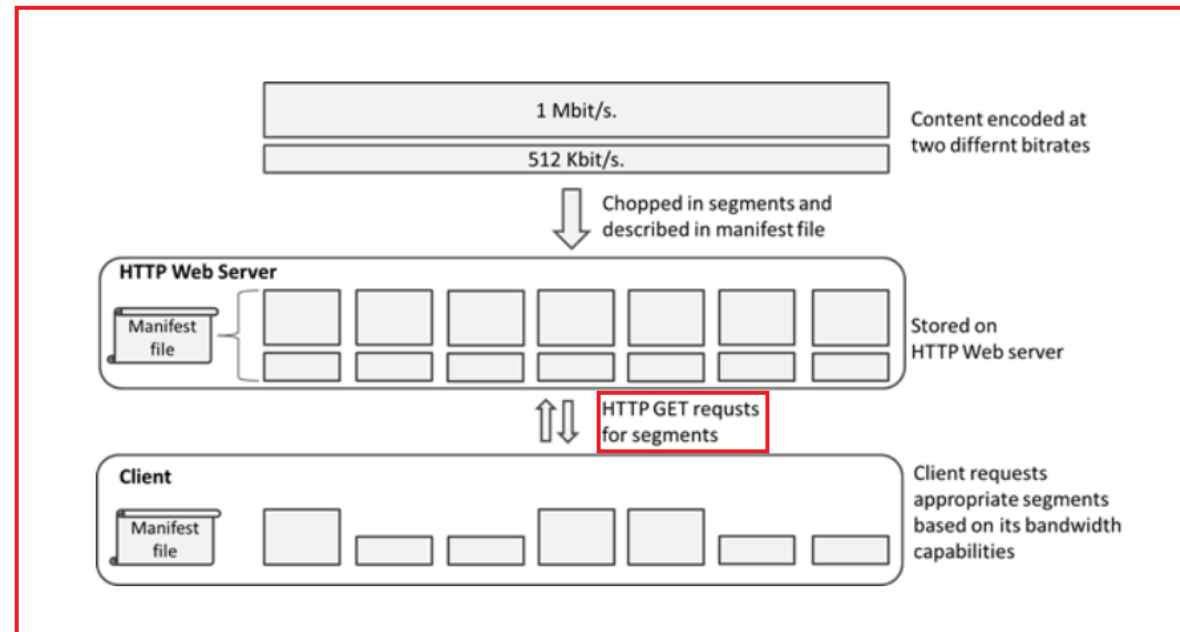


<https://developer.apple.com/videos/play/wwdc2012/502/>

creation of an HTTP GET request for requesting a fast search operation of an original video stream, the request stating a playback speed parameter and an initial position and optionally at least one

The Standard practices creation of an HTTP GET request (e.g., an HTTP request by the HLS client) for requesting a fast search operation (e.g., trick mode operation such as fast forward or rewind) of an original video stream (e.g., a video stream), the request stating a playback speed parameter (e.g., playout rate) and an initial position (e.g., start timing of a segment when the trick mode is requested) and optionally at least one parameter selected from a group of parameters consisting of file name, file type, path (e.g., URL of requested segments for trick mode operation), and playback direction.

parameter selected from a group of parameters consisting of file name, file type, path, and playback direction;




<http://blog.jpauli.tech/live-video-streaming-stack-from-home/>

As shown below, a client sends an HTTP Get request to an HLS media segment distribution server for media segments with URL of the media segments. Exemplary requests are shown below.

A typical HLS streaming session uses the following exchanges.

The client initiates streaming by sending an HTTP GET request to the server.

The server creates a playlist in an Extended M3U file encoded in UTF-8 (.m3u8). This playlist (index) is a manifest file that includes a set of URLs to media files (.ts segments) with their bitrates and sequence numbers.

 Copy

```
GET /vod/mp4:sample.mp4/playlist.m3u8 HTTP/1.1
Host: localhost:1935
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2272.101 Safari/
Accept: */*
Referer: http://localhost/jwenterprise/index.asp
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
Cookie: ASPSESSIONIDCQ8SBCRR=IAKNOGHJCJMDALEDFLDFFNDJH; ASPSESSIONIDASASBDQQ=CKDJNKHCFKAMNNFJDFPKOHC; DoNotShowFTU=

HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: no-cache
Date: Thu, 02 Apr 2015 17:01:00 GMT
Content-Type: application/vnd.apple.mpegurl
Content-Length: 137
```

<https://www.wowza.com/docs/how-to-troubleshoot-apple-hls-playback>

To stream an HLS-encoded video, make a request to an `m3u8` playlist, and set the `hls` attribute to `{}` to use the default options, or customize streaming options:

```
scenarios:
  - name: "Stream an HLS video"
    flow:
      - get:
        url: "/streams/xyz0123/xyz0123.m3u8"
        hls:
          concurrency: 2
          streamSelector:
            resolution:
              width: 320
              height: 184
          throttle: 100
```

<https://artillery.io/docs/guides/plugins/plugin-hls.html>

6.3.3. Playing the Media Playlist File

The client SHALL choose which Media Segment to play first from the Media Playlist when playback starts. If the EXT-X-ENDLIST tag is not present and the client intends to play the media normally, the client SHOULD NOT choose a segment that starts less than three target durations from the end of the Playlist file. Doing so can trigger playback stalls.

Normal playback can be achieved by playing the Media Segments in the order that they appear in the Playlist. The client MAY present the available media in any way it wishes, including normal playback, random access, and trick modes.

The encoding parameters for samples in a Media Segment and across multiple Media Segments in a Media Playlist SHOULD remain consistent. However, clients SHOULD deal with encoding changes as they are encountered, for example, by scaling video content to accommodate a resolution change. If the Variant Stream includes a RESOLUTION attribute, clients SHOULD display all video within a rectangle with the same proportions as that resolution.

<https://tools.ietf.org/html/rfc8216>

3. Media Segments

A Media Playlist contains a series of Media Segments that make up the overall presentation. A Media Segment is specified by a URI and optionally a byte range.

The duration of each Media Segment is indicated in the Media Playlist by its EXTINF tag ([Section 4.3.2.1](#)).

Each segment in a Media Playlist has a unique integer Media Sequence Number. The Media Sequence Number of the first segment in the Media Playlist is either 0 or declared in the Playlist ([Section 4.3.3.2](#)). The Media Sequence Number of every other segment is equal to the Media Sequence Number of the segment that precedes it plus one.

Each Media Segment MUST carry the continuation of the encoded bitstream from the end of the segment with the previous Media Sequence Number, where values in a series such as timestamps and Continuity Counters MUST continue uninterrupted. The only exceptions are the first Media Segment ever to appear in a Media Playlist and Media Segments that are explicitly signaled as discontinuities ([Section 4.3.2.3](#)). Unmarked media discontinuities can trigger playback errors.

Any Media Segment that contains video SHOULD include enough information to initialize a video decoder and decode a continuous set of frames that includes the final frame in the Segment; network efficiency is optimized if there is enough information in the Segment to decode all frames in the Segment. For example, any Media Segment containing H.264 video SHOULD contain an Instantaneous Decoding Refresh (IDR); frames prior to the first IDR will be downloaded but possibly discarded.

<https://tools.ietf.org/html/rfc8216>

6.3.5. Determining the Next Segment to Load

The client **MUST** examine the Media Playlist file every time it is loaded or reloaded to determine the next Media Segment to load, as the set of available media **MAY** have changed.

The first segment to load is generally the segment that the client has chosen to play first (see [Section 6.3.3](#)).

In order to play the presentation normally, the next Media Segment to load is the one with the lowest Media Sequence Number that is greater than the Media Sequence Number of the last Media Segment loaded.

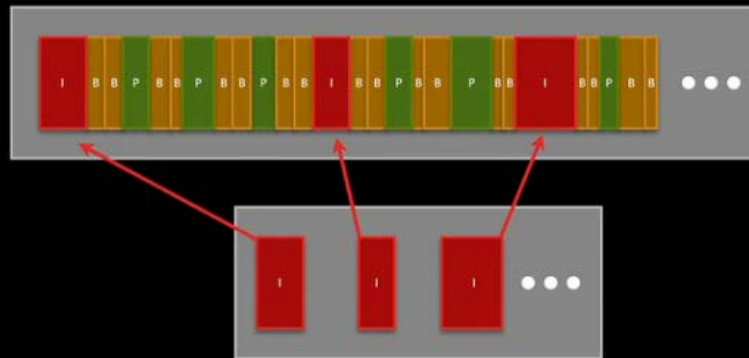
<https://tools.ietf.org/html/rfc8216>

Fast Forward and Reverse Playback

- Uses I-frame playlists
- Essential for AirPlay and AppleTV
- The higher the I-frame frequency, the smoother the playback

I-frame Playlists

Can reuse existing media files



▶ 24:03 / 41:03



<https://developer.apple.com/videos/play/wwdc2012/502/>

As shown below, a client sends an HTTP Get request to an HLS media segment distribution server for media segments with video playback at a higher speed than normal speed. Exemplary requests are shown below.

```

<html>
  <head>

    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
    <meta http-equiv="content-type" content="text/html; charset=utf-8">
  </head>

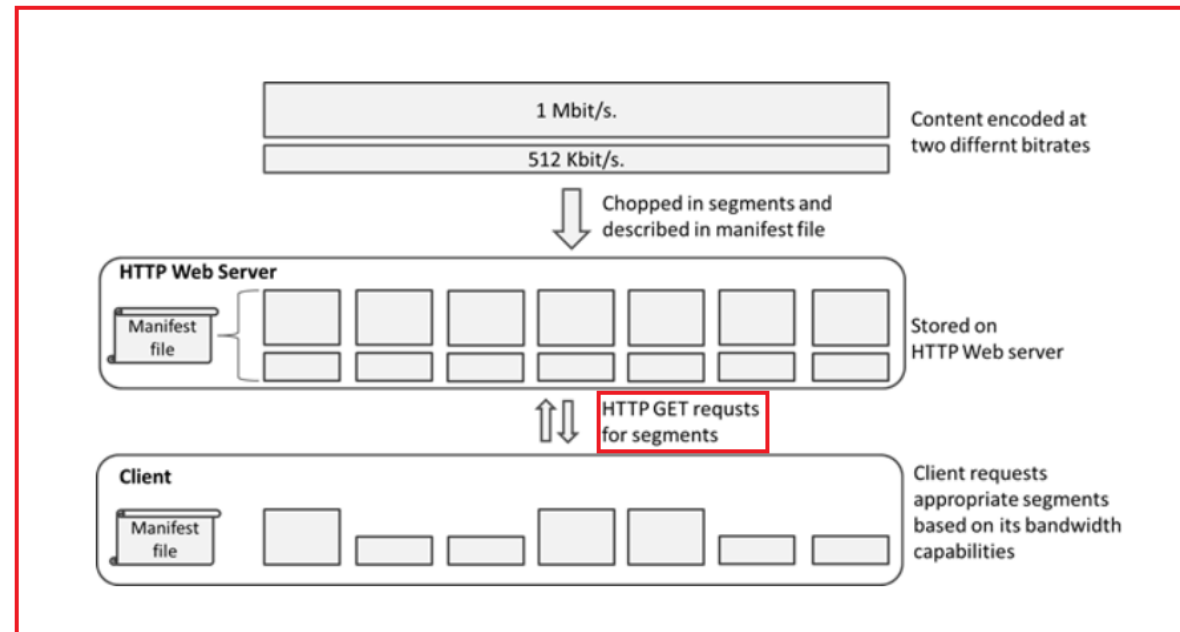
  <body>
    <Button onclick = "getPlaySpeed ()" type = "button"> playback speed is how much? </ Button>
    <Button onclick = "setPlaySpeed ()" type = "button"> to the video player to quickly </ button>
  <br />
  <br />
  <video id="video" controls="controls">
    <source src="http://www.streambox.fr/playlists/test_001/stream.m3u8" type="video/mp4" >
  </video>
  <script>
    var video = document.getElementById("video");
    function getPlaySpeed() {
      alert (video.playbackRate); // get video / audio playback rate
    }
    function setPlaySpeed() {
      video.playbackRate = 2; // set the video / audio playback speed
    }
  </script>
</body>
</html>

```

<https://programmersought.com/article/32112931107/>

transmission of the HTTP GET request to a source appliance; and

The Standard practices transmission of the HTTP GET request (e.g., the HTTP request by the HLS client device) to a source appliance (e.g., HLS server).




<http://blog.jpauli.tech/live-video-streaming-stack-from-home/>

As shown below, a client sends an HTTP Get request to an HLS media segment distribution server for media segments with URL of the media segments. Exemplary requests are shown below.

A typical HLS streaming session uses the following exchanges.

The client initiates streaming by sending an HTTP GET request to the server.

The server creates a playlist in an Extended M3U file encoded in UTF-8 (.m3u8). This playlist (index) is a manifest file that includes a set of URLs to media files (.ts segments) with their bitrates and sequence numbers.

 Copy

```
GET /vod/mp4:sample.mp4/playlist.m3u8 HTTP/1.1
Host: localhost:1935
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2272.101 Safari/
Accept: */*
Referer: http://localhost/jwenterprise/index.asp
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
Cookie: ASPSESSIONIDCQ8SBCRR=IAKNOGHCMJDALEDFLDFFNDJH; ASPSESSIONIDASASBDQQ=CKDJNKHCFKAMNNFJDFPKOHC; DoNotShowFTU=

HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: no-cache
Date: Thu, 02 Apr 2015 17:01:00 GMT
Content-Type: application/vnd.apple.mpegurl
Content-Length: 137
```

<https://www.wowza.com/docs/how-to-troubleshoot-apple-hls-playback>

To stream an HLS-encoded video, make a request to an `m3u8` playlist, and set the `hls` attribute to `{}` to use the default options, or customize streaming options:

```
scenarios:
  - name: "Stream an HLS video"
    flow:
      - get:
        url: "/streams/xyz0123/xyz0123.m3u8"
        hls:
          concurrency: 2
          streamSelector:
            resolution:
              width: 320
              height: 184
          throttle: 100
```

<https://artillery.io/docs/guides/plugins/plugin-hls.html>

6.3.3. Playing the Media Playlist File

The client SHALL choose which Media Segment to play first from the Media Playlist when playback starts. If the EXT-X-ENDLIST tag is not present and the client intends to play the media normally, the client SHOULD NOT choose a segment that starts less than three target durations from the end of the Playlist file. Doing so can trigger playback stalls.

Normal playback can be achieved by playing the Media Segments in the order that they appear in the Playlist. The client MAY present the available media in any way it wishes, including normal playback, random access, and trick modes.

The encoding parameters for samples in a Media Segment and across multiple Media Segments in a Media Playlist SHOULD remain consistent. However, clients SHOULD deal with encoding changes as they are encountered, for example, by scaling video content to accommodate a resolution change. If the Variant Stream includes a RESOLUTION attribute, clients SHOULD display all video within a rectangle with the same proportions as that resolution.

<https://tools.ietf.org/html/rfc8216>

3. Media Segments

A Media Playlist contains a series of Media Segments that make up the overall presentation. A Media Segment is specified by a URI and optionally a byte range.

The duration of each Media Segment is indicated in the Media Playlist by its EXTINF tag ([Section 4.3.2.1](#)).

Each segment in a Media Playlist has a unique integer Media Sequence Number. The Media Sequence Number of the first segment in the Media Playlist is either 0 or declared in the Playlist ([Section 4.3.3.2](#)). The Media Sequence Number of every other segment is equal to the Media Sequence Number of the segment that precedes it plus one.

Each Media Segment MUST carry the continuation of the encoded bitstream from the end of the segment with the previous Media Sequence Number, where values in a series such as timestamps and Continuity Counters MUST continue uninterrupted. The only exceptions are the first Media Segment ever to appear in a Media Playlist and Media Segments that are explicitly signaled as discontinuities ([Section 4.3.2.3](#)). Unmarked media discontinuities can trigger playback errors.

Any Media Segment that contains video SHOULD include enough information to initialize a video decoder and decode a continuous set of frames that includes the final frame in the Segment; network efficiency is optimized if there is enough information in the Segment to decode all frames in the Segment. For example, any Media Segment containing H.264 video SHOULD contain an Instantaneous Decoding Refresh (IDR); frames prior to the first IDR will be downloaded but possibly discarded.

<https://tools.ietf.org/html/rfc8216>

6.3.5. Determining the Next Segment to Load

The client **MUST** examine the Media Playlist file every time it is loaded or reloaded to determine the next Media Segment to load, as the set of available media **MAY** have changed.

The first segment to load is generally the segment that the client has chosen to play first (see [Section 6.3.3](#)).

In order to play the presentation normally, the next Media Segment to load is the one with the lowest Media Sequence Number that is greater than the Media Sequence Number of the last Media Segment loaded.

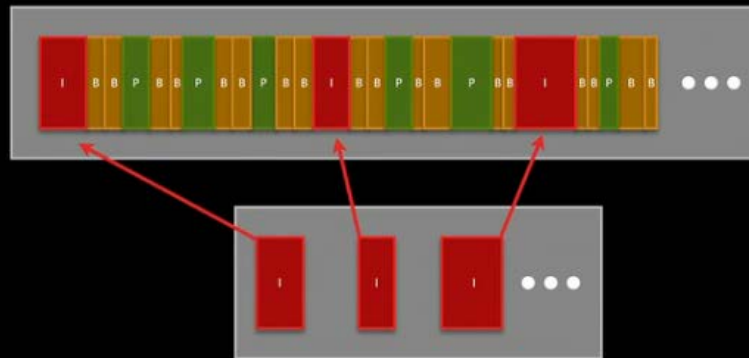
<https://tools.ietf.org/html/rfc8216>

Fast Forward and Reverse Playback

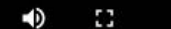
- Uses I-frame playlists
- Essential for AirPlay and AppleTV
- The higher the I-frame frequency, the smoother the playback

I-frame Playlists

Can reuse existing media files



▶ 24:03 / 41:03



<https://developer.apple.com/videos/play/wwdc2012/502/>

As shown below, a client sends an HTTP Get request to an HLS media segment distribution server for media segments with video playback at a higher speed than normal speed. Exemplary requests are shown below.

```

<html>
  <head>

    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
    <meta http-equiv="content-type" content="text/html; charset=utf-8">
  </head>

  <body>
    <Button onclick = "getPlaySpeed ()" type = "button"> playback speed is how much? </ Button>
    <Button onclick = "setPlaySpeed ()" type = "button"> to the video player to quickly </ button>
  <br />
  <br />
  <video id="video" controls="controls">
    <source src="http://www.streambox.fr/playlists/test_001/stream.m3u8" type="video/mp4" >
  </video>
  <script>
    var video = document.getElementById("video");
    function getPlaySpeed() {
      alert (video.playbackRate); // get video / audio playback rate
    }
    function setPlaySpeed() {
      video.playbackRate = 2; // set the video / audio playback speed
    }
  </script>
</body>
</html>

```

<https://programmersought.com/article/32112931107/>

discontinuous transmission, in sections, of selected video frames of an original encoded

The Standard practices discontinuous transmission (e.g., discontinues transmission of media segments during a trick mode operation such as fast-forward, fast-rewind, etc.), in sections (e.g., media segments), of selected video frames (e.g., selected video frames in trick mode which skips other frames and only plays I frames) of an original encoded video stream from the source appliance (e.g., HTTP server) to a destination appliance (e.g., HLS client) in

<p>video stream from the source appliance to a destination appliance in a HTTP response using an extended HTTP chunked transfer encoding mode, in which the selected encoded video frames for the fast search operation are transported in respective chunks, wherein each chunk includes one complete respective selected encoded video frame in a second part and information about a starting time, as located in the original encoded video stream, of the respective selected video frame in a first part, wherein the second part is different from the first part and the information about a starting time of the respective selected video frame being positioned in a commentary line of the first part.</p>	<p>an HTTP response using an extended HTTP chunked transfer encoding mode (e.g., transferring video data in segments or chunks in encoding format), in which the selected encoded video frames for the fast search operation (e.g., trick mode operation such as fast forward or rewind) are transported in respective chunks (e.g., segments or chunks), wherein each chunk includes one complete respective selected encoded video frame (e.g., a segment that comprises an I-frame) in a second part (e.g., the media segment) and information about a starting time (e.g., media segment timestamp), as located in the original encoded video stream (media segments for trick mode are time-aligned with the segments in the original media streams, e.g., the original video stream, which enables switching between the trick mode stream and the normal play rate streams), of the respective selected video frame (e.g., the selected video data frame in trick mode) in a first part (e.g., media segment details such as media segment timestamp, sequence no., etc.), wherein the second part (e.g., the media segment) is different from the first part (e.g., media segment details such as media segment timestamp, sequence no., etc.) and the information about a starting time of the respective selected video frame being positioned in a commentary line of the first part (e.g., media segment details such as media segment timestamp, sequence no., etc.).</p>
--	--

6.3.3. Playing the Media Playlist File

The client SHALL choose which Media Segment to play first from the Media Playlist when playback starts. If the EXT-X-ENDLIST tag is not present and the client intends to play the media normally, the client SHOULD NOT choose a segment that starts less than three target durations from the end of the Playlist file. Doing so can trigger playback stalls.

Normal playback can be achieved by playing the Media Segments in the order that they appear in the Playlist. The client MAY present the available media in any way it wishes, including normal playback, random access, and trick modes.

The encoding parameters for samples in a Media Segment and across multiple Media Segments in a Media Playlist SHOULD remain consistent. However, clients SHOULD deal with encoding changes as they are encountered, for example, by scaling video content to accommodate a resolution change. If the Variant Stream includes a RESOLUTION attribute, clients SHOULD display all video within a rectangle with the same proportions as that resolution.

<https://tools.ietf.org/html/rfc8216>

Media Segments

A Media Playlist contains a series of Media Segments that make up the overall presentation. A Media Segment is specified by a URI and optionally a byte range.

The duration of each Media Segment is indicated in the Media Playlist by its EXTINF tag ([Section 4.3.2.1](#)).

Each segment in a Media Playlist has a unique integer Media Sequence Number. The Media Sequence Number of the first segment in the Media Playlist is either 0 or declared in the Playlist ([Section 4.3.3.2](#)). The Media Sequence Number of every other segment is equal to the Media Sequence Number of the segment that precedes it plus one.

Each Media Segment MUST carry the continuation of the encoded bitstream from the end of the segment with the previous Media Sequence Number, where values in a series such as timestamps and Continuity Counters MUST continue uninterrupted. The only exceptions are the first Media Segment ever to appear in a Media Playlist and Media Segments that are explicitly signaled as discontinuities ([Section 4.3.2.3](#)). Unmarked media discontinuities can trigger playback errors.

Any Media Segment that contains video SHOULD include enough information to initialize a video decoder and decode a continuous set of frames that includes the final frame in the Segment; network efficiency is optimized if there is enough information in the Segment to decode all frames in the Segment. For example, any Media Segment containing H.264 video SHOULD contain an Instantaneous Decoding Refresh (IDR); frames prior to the first IDR will be downloaded but possibly discarded.

<https://tools.ietf.org/html/rfc8216>

To play this Playlist, the client first downloads it and then downloads and plays each Media Segment declared within it. The client reloads the Playlist as described in this document to discover any added segments. Data SHOULD be carried over HTTP [RFC7230], but, in general, a URI can specify any protocol that can reliably transfer the specified resource on demand.

A more complex presentation can be described by a Master Playlist. A Master Playlist provides a set of Variant Streams, each of which describes a different version of the same content.

A Variant Stream includes a Media Playlist that specifies media encoded at a particular bit rate, in a particular format, and at a particular resolution for media containing video.

<https://tools.ietf.org/html/rfc8216>

Each Media Segment MUST carry the continuation of the encoded bitstream from the end of the segment with the previous Media Sequence Number, where values in a series such as timestamps and Continuity Counters MUST continue uninterrupted. The only exceptions are the first Media Segment ever to appear in a Media Playlist and Media Segments that are explicitly signaled as discontinuities (Section 4.3.2.3). Unmarked media discontinuities can trigger playback errors.

The server MAY associate an absolute date and time with a Media Segment by applying an EXT-X-PROGRAM-DATE-TIME tag to it. This defines an informative mapping of the (wall-clock) date and time specified by the tag to the first media timestamp in the segment, which may be used as a basis for seeking, for display, or for other purposes. If a server provides this mapping, it SHOULD apply an EXT-X-PROGRAM-DATE-TIME tag to every segment that has an EXT-X-DISCONTINUITY tag applied to it.

The server MUST meet the following constraints when producing Variant Streams in order to allow clients to switch between them seamlessly:

- o Each Variant Stream MUST present the same content.
- o Matching content in Variant Streams MUST have matching timestamps. This allows clients to synchronize the media.

<https://tools.ietf.org/html/rfc8216>

The server sends the client a chunklist, which is a subset of the files in the playlist.

Copy

```
GET /vod/mp4:sample.mp4/chunklist_w1556499301.m3u8 HTTP/1.1
Host: localhost:1935
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2272.101 Safari/
Accept: */*
Referer: http://localhost/jwenterprise/index.asp
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
Cookie: ASPSESSIONIDCQBSBCRR=IAKNOGHJCJMDALEDFLDFFNDJH; ASPSESSIONIDASASBDQQ=CKDJNKHCFKAMNNFJDFPKOHC; DoNotShowFTU=

HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: no-cache
Date: Thu, 02 Apr 2015 17:01:01 GMT
Content-Type: application/vnd.apple.mpegurl
Content-Length: 2247
```

<https://www.wowza.com/docs/how-to-troubleshoot-apple-hls-playback>